KERJA PRAKTIK – IF184801

Pengembangan Microservice NPS Fabelio untuk Meningkatkan Kehandalan Microservices

Fabelio
Jl. Barito II No.56, RT.4/RW.4, Kramat Pela, Kec. Kby. Baru, Kota Jakarta Selatan, Daerah Khusus IbukotaJakarta 12130
Periode: 11 Januari 2021 – 11 April 2021

Oleh:
Yovi Agustian 05111740000125

Pembimbing Jurusan
Hudan Studiawan, S.Kom., M.Kom.,Ph.D.
Pembimbing Lapangan
Muhammad Ilyas Dwi Jayanto

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas

[Halaman ini sengaja dikosongkan]

KERJA PRAKTIK – IF184801

Pengembangan Microservice NPS Fabelio untuk Meningkatkan Kehandalan Microservices

Fabelio
Jl. Barito II No.56, RT.4/RW.4, Kramat Pela, Kec. Kby. Baru, Kota Jakarta Selatan, Daerah Khusus IbukotaJakarta 12130
Periode: 11 Januari 2021 – 11 April 2021

Oleh:
Yovi Agustian 05111740000125

Pembimbing Jurusan
Hudan Studiawan, S.Kom., M.Kom.,Ph.D.
Pembimbing Lapangan
Muhammad Ilyas Dwi Jayanto

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember Surabaya
2021

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN
KERJA PRAKTIK

Pengembangan Microservice NPS Fabelio untuk Meningkatkan Kehandalan Microservices

Oleh:

Yovi Agustian 05111740000125

Mengetahui, Pembimbing Lapangan Kerja Praktik	Menyetujui, Dosen Pembimbing Kerja Praktik
	
M Ilyas Dwi Jayanto Software Engineer Fabelio	Hudan Studiawan, S.Kom., M.Kom., Ph.D. NIP. 198705112012121003

SURABAYA
April, 2021

[Halaman ini sengaja dikosongkan]

Pengembangan Microservice NPS Fabelio untuk Meningkatkan Kehandalan Microservices

Nama	**: Yovi Agustian**
NRP	**: 05111740000125**
Departemen	**: Informatika FTEIC**
Pembimbing Dep.	**: Hudan Studiawan, S.Kom., M.Kom.,Ph.D.**
Pembimbing Lap.	**: M Ilyas Dwi Jayanto**

ABSTRAK

Fabelio adalah perusahaan nasional di Indonesia yang bergerak di bidang furniture dan desain interior baik online maupun offline yang berfokus pada loose furniture rumah minimalis, modern dan perencanaan serta pengerjaan desain interior baik untuk rumah, apartemen, hingga kantor.

Dalam mendukung perkembangan bisnis yang begitu cepat maka diperlukan sistem yang handal sehingga dapat memperlancar kegiatan bisnis, oleh karena itu Fabelio membuat sistem yang bertujuan untuk mengotomasi kegiatan bisnis seperti integrasi stock, order, pembayaran, dan lain - lain. Namun karena padatnya trafic yang terjadi sering kali menyebabkan sistem melambat, bahkan down dan tidak dapat digunakan.

Maka dari itu Fabelio membuat sebuah tim khusus untuk mengatasi masalah ini yaitu tim NPS. Tim NPS membagi kegiatan – kegiatan bisnis menjadi beberapa microservice seperti microservice marvel untuk integrasi order, PMS untuk mengurus produk, Morbius untuk tracking estimasi biaya pengiriman, dan lain – lain. Diharapkan dengan cara ini dapat menghasilkan sistem yang handal yang dapat mendukung seluruh kegiatan bisnis dengan cepat sesuai dengan perkembangan Fabelio.

Kata kunci: Microservice, API, Server, Client

KATA PENGANTAR

Puji syukur saya sampaikan kepada Tuhan yang Maha Esa karena berkat rahmat-Nya saya dapat melaksanakan salah satu kewajiban saya sebagai mahasiswa Departemen Teknik Informatika, yakni Kerja Praktek (KP).

Saya menyadari masih ada kekurangan baik dalam pelaksanaan kerja praktik maupun penyusunan buku laporan ini. Namun, saya berharap buku laporan ini dapat menambah wawasan pembaca dan dapat menjadi sumber referensi. Saya mengharapkan kritik dan saran yang membangun untuk kesempurnaan buku laporan kerja praktik ini.

Melalui buku ini, saya juga ingin menyampaikan rasa terima kasih kepada orang-orang yang telah membantu, baik langsung maupun tidak langsung, dalam pelaksanaan kerja praktik hingga penyusunan laporan. Orang-orang tersebut antara lain adalah:

1. Kedua orang tua penulis karena telah memberikan dukungan serta membiayai penulis saat mengerjakan kerja praktik.
2. Bapak Hudan Studiawan, S.Kom., M.Kom.,Ph.D. selaku dosen pembimbing Kerja Praktik.
3. Bapak Ary Mazharuddin Shiddiqi, S.Kom., M.Sc. selaku koordinator Kerja Praktik.
4. M ilyas Dwi Jayanto, selaku pembimbing lapangan di team NPS Fabelio.
5. Glenn, selaku leader di team NPS Fabelio.
6. Rekan-rekan saya yang berada dalam tim NPS.
7. Rekan-rekan lain yang tidak bisa saya sebutkan satu per satu.

Surabaya, 26 April 2021

Yovi Agustian

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1. Latar Belakang

Saat ini indonesia telah memasuki era industri digital yang dapat diamati dari maraknya perkembangan teknologi dan internet di Indonesia. Hal ini menyebabkan perubahan pola konsumsi masyarakat indonesia sendiri, dari belanja secara konvensional berubah menjadi berbelanja secara online sehingga untuk mengikuti demand dari masyarakat ini banyak bermunculan platform *e-commerce* baru di Indonesia dan Fabelio merupakan salah satunya.

Fabelio adalah salah satu perusahaan *e-commerce* yang ada di Indonesia yang bergerak di bidang furniture. Dalam melakukan pemasaran, perusahaan ini menggunakan strategi multi channel dengan membangun website (www.fabelio.com) yang berfungsi sebagai etalase dari produk yang ditawarkan sekaligus mempermudah dalam melakukan kegiatan transaksi jual – beli.

Saat ini ini website Fabelio telah memiliki *traffic* yang sangat besar dan terkadang menyebabkan website menjadi lambat saat diakses. Maka dari itu dalam menjaga kenyamanan pengguna saat mencari produk atau bertransaksi lewat website yang dimilikinya, Fabelio selalu berusaha memberikan pelayanan yang terbaik contohnya dengan menjaga kecepatan website dan konsistensi data yang ditampilkannya. Untuk mewujudkannnya Fabelio membuat tim yang bernama NPS untuk menjawab kebutuhan tersebut. Tim NPS adalah tim yang bertanggung jawab dalam menjaga kecepatan integrasi dan konsistensi data dengan selalu melakukan pengembangan dan perbaikan setiap *microservice* yang berkaitan.

Penulis sebagai mahasiswa tingkat akhir mendapatkan kesempatan untuk melakukan KP (kerja praktek) di Fabelio khususnya bergabung menjadi bagian dari tim NPS. Dengan demikian penulis diharapkan mampu mengaplikasikan ilmu – ilmu yang telah didapat selama masa perkuliahan ke dunia nyata.

1.2. Tujuan

Tujuan dari kerja praktek ini adalah untuk menyelesaikan kewajiban kuliah yakni KP (kerja praktik) di Institut Teknologi Sepuluh Nopember dengan beban dua SKS. Selain itu penulis juga bertujuan untuk merasakan dan belajar langsung di dunia kerja dengan para praktisi serta mencari pengalaman baru yang belum pernah didapat sebelumnya. Tidak lupa juga untuk berkontribusi dengan menerapkan ilmu – ilmu yang telah didapat selama masa perkuliahan ke dunia nyata.

1.3. Manfaat

Manfaat dari pengembangan dan perawatan *microservice* ini adalah untuk membantu Fabelio dalam menyelesaikan kegiatan bisnis yang dimilikinya sehingga dapat terus berkembang dan mencapai misinya untuk mewujudkan surga pada hunian setiap penggunanya dan mengembangkan sektor furniture di Indonesia.

1.4. Rumusan Masalah

Berikut ini rumusan masalah pada kerja praktik pengembangan *microservice* di Fabelio:

1. Bagaimana cara mengoptimasi pencarian data?
2. Bagaimana mengintegrasi data antar *microservice*?
3. Bagaimana mengatasi error yang terjadi di *microservice*?
4. Bagaimana mendokumentasi API?
5. Bagaimana mengontrol *microservice*?

1.5. Lokasi dan Waktu Kerja Praktik

Kerja praktik ini dilaksanakan pada waktu dan tempat sebagai berikut:

Lokasi : Fabelio
Bidang : NPS
Alamat : Jl. Barito II No.56, RT.4/RW.4, Kramat Pela, Kec. Kby. Baru, Kota Jakarta Selatan, Daerah Khusus Ibukota
Waktu : 11 Januari – 11 April 2021
Hari Kerja : Senin-Jumat
Jam Kerja : 09.00 WIB - 18.00 WIB (*Full time*)

1.6. Metodologi Kerja Praktik

Tahapan pengerjaan kerja praktik dapat dijabarkan sebagai berikut:

1. Perumusan Masalah

Untuk mengetahui permasalahan apa yang harus diselesaikan serta diberikan penjelasan mengenai alasan mengapa tim NPS beserta *microservice* yang menjadi tanggung jawabnya dibutuhkan. Dijelaskan juga secara rinci mengenai bagaimana alur *microservice* itu akan berjalan dan terimplementasikan untuk mengerjakan proses bisnis yang ada. Penjelasan mengenai hal ini dijelaskan oleh pembimbing lapangan pada saat kerja praktik.

2. Studi Literatur

Pada tahap ini merupakan proses pembelajaran mengenai *microservice* dan *tools* yang digunakan pada setiap *microservice*. Pada tahap ini pula dilakukan proses pencarian informasi serta pemahaman mengenai berbagai macam *library* yang akan digunakan. Informasi didapat dari dokumentasi yang telah dibuat oleh tim NPS Fabelio serta informasi tambahan dari internet.

3. Analisis dan Perancangan

Tahap ini meliputi analisis lebih dalam mengenai tentang apa saja yang akan dibuat dalam pengembangan *microservice*. Langkah-langkah konkrit yang digunakan dalam tahap ini adalah sebagai berikut:

a) Memahami kebutuhan yang diperlukan untuk pengembangan.
b) Mempelajari dan memahami *flow* dan arsitektur yang diterapkan pada *microservice*.
c) Memahami Node.js sebagai *framework* yang akan digunakan dlaam melakukan pengembangan.

4. **Implementasi Sistem**

Pengembangan dilakukan selama kurang lebih tiga bulan untuk dengan tipe *Sprint*. Segala jenis pengembangan dan pengerjaan yang dilakuan tidak terlepas dari *review* oleh satu tim, sehingga kode sumber yang sudah terverifikasi oleh tiga atau lebih anggota tim hingga dapat digabungkan dalam kode sumber utama atau *master code*.

5. **Pengujian dan Evaluasi**

Pengujian dilakukan dengan menggunakan metode *Black Box* dan *Mocking*. Penguji menguji semua alternatif kemungkinan yang dapat terjadi pada unit testing untuk meningkatkan *coverage* dan juga dilakukan *mocking* untuk *external end point* dengan menggunakan Sinon, Mocha, dan Chai.

6. **Kesimpulan dan Saran**

Pada bab ini akan dipaparkan kesimpulan yang dapat diambil dari keseluruhan proses kerja praktik yang telah dilakukan dan juga saran - saran yang dapat diberikan selama penulis melakukan pengerjaan kerja praktik.

1.7. Sistematika Laporan

Laporan kerja praktik ini terdiri dari enam bab dengan rincian sebagai berikut:

1. **Bab I Pendahuluan**

Pada bab ini dijelaskan tentang latar belakang permasalahan, tujuan, waktu pelaksanaan, serta sistematika pengerjaan kerja praktik dan juga penulisan laporan kerja praktik.

2. **Bab II Profil Perusahaan**

Pada bab ini, dijelaskan secara rinci tentang profil perusahaan saat melaksanakan kerja praktik, yakni Fabelio.

3. **Bab III Tinjauan Pustaka**

Pada bab ini, dijelaskan mengenai tinjauan pustaka dan literatur yang digunakan dalam penyelesaian kerja praktik.

4. **Bab IV Implementasi Sistem**

Pada bab ini, berisi penjelasan tahap-tahap yang dilakukan untuk proses pengembangan aplikasi berbasis *microservice* menggunakan Node.js dan React.

5. **Bab V Pengujian dan Evaluasi**

Pada bab ini, dijelaskan tentang hasil pengujian dan evaluasi dari sistem yang telah dikembangkan selama pelaksanaan kerja praktik.

6. **Bab VI Kesimpulan dan Saran**

Pada bab ini, dipaparkan kesimpulan yang dapat diambil dan juga saran selama pengerjaan kerja praktik.

[Halaman ini sengaja dikosongkan]

BAB II
PROFIL PERUSAHAAN

2.1. Sejarah Perusahaan

PT. Tiga Elora Nusantara merupakan salah satu perusahaan nasional yang bergerak di bidang furnitur dengan mengedepankan sistem *E-Procurement.* Fabelio merupakan suatu merek furnitur yang dimiliki oleh PT. Tiga Elora Nusantara dalam menjalani bisnis dibidang ini. Perusahaan ini didirikan oleh Krishnan Menon dan Christian Sutardi dengan melihat kebutuhan akan furnitur di Indonesia yang semakin berkembang.

Hingga saat ini PT Tiga Elora Nusantara telah memiliki 15 showroom furniture yang bertujuan untuk menjangkau konsumen yang ingin melihat secara langsung produk yang ditawarkan oleh PT Tiga Elora Nusantara (Fabelio). Perusahaan ini memiliki domisili di Kawasan Industri dan Pergudangan Taman Tekno BSD City dalam hal produksi dan penyimpanan barang, serta memiliki domisili di Jl.Panglima Polim Jakarta Selatan dalam hal penjualan barang.

Furnitur yang dihasilkan terbuat dari ragam jenis kayu berkualitas yang sudah melewati proses produksi oleh ahlinya. Selain bekerjasama dengan pengrajin dan penghasil furniture terbaik dengan desainer furnitur terbaik Indonesia untuk menghasilkan produk yang orisinil, inovatif, dan otentik.

Perusahaan ini adalah kumpulan professional yang telah berkecimpung di dalam dunia furnitur selama lebih dari 20 tahun. Dimana terdapat orang-orang yang paling berkontribusi besar di dalam industry desain di Indonesia. Produk yang dihasilkan oleh perusahaan ini dibuat sesuai permintaan dengan material yang sudah teruji oleh pemasok maupun pengrajin yang telah diseleksi dengan standar jauh diatas rata-rata.

2.2. Profil Perusahaan Fabelio

Fabelio.com adalah suatu portal web dalam bidang furnitur yang menghubungkan antara pembeli dan penjual beragam furnitur berkualitas tinggi. Produk- produk yang dijual oleh

penjual melalui Fabelio.com adalah furnitur berkualitas premium dengan harga terbaik. Sejak didirikan pada tahun 2015, hingga saat ini Fabelio telah berkembang dengan pesat baik online maupun offline dan telah memiliki 15 showroom furniture tidak hanya di Indonesia tetapi juga di Asia Tenggara.

Produk furnitur yang tersedia di Fabelio adalah murni hasil karya anak bangsa yang dibuat dengan material kayu Indonesia berkualitas premium. Berawal dari keinginan untuk menjadikan setiap rumah layak huni, Fabelio mewujudkannya melalui desain-desain unik yang dikerjakan oleh pengrajin lokal.

Setiap produk yang ditawarkan oleh Fabelio memiliki kualitas setransparan harga yang perlu dibayar. Fabelio bekerjasama dengan pengrajin-pengrajin luar biasa. Bagi Fabelio, proses ini dapat dilakukan tanpa perantara yang melonjakan harga produksi, sehingga customer dapat menikmati produk berkualitas tanpa tarif yang berlebihan.

Memiliki ciri khas tertentu merupakan hal terpenting di Fabelio sehingga kami berdedikasi tinggi untuk menciptakan desain furnitur yang unik secara visual, tetapi tetap fungsional dan mengutamakan kebutuhan pelanggan.

Fabelio percaya bahwa setiap orang, berhak mendapatkan desain dan kualitas furnitur yang memuaskan, di mana pun berada, kapan pun dinginkan. Karena itulah Fabelio mengunggulkan kemudahan bagi penggunanya untuk mengakses produk idaman dengan sangat mudah lewat website dan showroom yang ada.

2.3. Visi dan Misi Perusahaan

PT Fabelio memiliki Misi yang berfungsi untuk menunjang Visinya sebagai berikut:

2.3.1 Visi

Menjadi sebuah wadah berupa portal web dalam bidang furnitur yang menghubungkan pembeli dan penjual beragam furnitur berkualitas tinggi dengan harga terbaik, yang dimana setiap keluarga dapat menemukan suatu kebahagiaan dan surga dalam rumah dengan menyediakan produk furnitur dan konsultan desain. Fabelio.com

berkomitmen untuk selalu memberikan hasil 100% dan memberikan hasil rancangan dan produksi dari anak negeri.

2.3.2 Misi

Menciptakan surga di hunian selain itu fabelio juga memiliki misi untuk membantu dan memajukan komunitas industri pengrajin di Indonesia.

[Halaman ini sengaja dikosongkan]

BAB III
TINJAUAN PUSTAKA

Pada bab ini, akan dijelaskan mengenai dasar teori yang digunakan selama proses kerja praktik.

3.1 Microservice

Microservices adalah suatu *framework architecture* yang dipakai sebagai model dalam pembuatan aplikasi *cloud* yang modern. Didalam *microservices* setiap aplikasi dibangun sebagai sekumpulan *service* dan setiap layanan berjalan dalam prosesnya sendiri. Masing-masing dari aplikasi tersebut saling berkomunikasi melalu API *(Application Programing Interface*).

3.2 API

API atau *Application Programming Interface* adalah sebuah *interface* yang dapat menghubungkan aplikasi satu dengan aplikasi lainnya. API berperan sebagai perantara antar berbagai aplikasi berbeda, baik dalam satu *platform* yang sama atau lintas *platform*. Dalam pengembangan ini API berguna untuk menghubungkan antar *microservice*.

3.3 Datadog

Datadog adalah layanan *monitoring* untuk aplikasi dengan skala *cloud*, menyediakan pemantauan server, database, *tools*, dan *services*, melalui *platform* analisis data berbasis SaaS (*Software as a Service*). Datadog menggunakan agen berbasis Golang, yang ditulis ulang dari bahas Python. *Backend*-nya dibangun menggunakan sejumlah teknologi seperti D3, Apache Cassandra, Kafka, PostgreSQL. Data ditampilkan dengan antarmuka yang aktif dan *real-time* yang dapat disesuaikan dengan kebutuhan spesifik tim (Datadoghq).

3.4 MongoDB

MySQL MongoDB adalah salah satu jenis database yang

menggunakan konsep NoSQL berbasis dokumen. Hal ini tentu berbeda dengan database yang menggunakan konsep MySQL dengan RDBMSnya. Keunggulan dari MongoDB ini adalah dalam sistem penyimpanan data tidak lagi menggunakan tabel. Akan tetapi, menggunakan dokumen terstruktur layaknya JSON sebab telah menggunakan JavaScript. Sehingga performa yang dihasilkan oleh MongoDB akan lebih cepat sebab juga didukung oleh memcached.

3.5 Node.js

Node.js adalah adalah *platform* buatan Ryan Dahl yang dikenalkan pada tahun 2009. Node.js merupakan perangkat lunak yang didesain untuk mengembangkan aplikasi berbasis web dan ditulis dalam sintaks bahasa pemrograman JavaScript. Pengguna dapat menjalankan JavaScript dari sisi server dengan Node.js. yang bertugas untuk mengeksekusi kode JavaScript sebelum halaman website ditampilkan di browser. Disamping itu, Node.js juga memiliki pustaka server sendiri sehingga tidak perlu menggunakan program server web seperti Nginx dan Apache. Dengan model *event-driven* dan *non-blocking* I/O-nya, Node.js lebih mampu menangani banyak proses secara bersamaan daripada platform bersifat *thread-based networking*.

3.6 JavaScript

Javascript adalah bahasa pemrograman *client-side* yang sering digunakan oleh pengembang web bersamaan dengan HTML dan CSS untuk membuat halaman website yang bersifat interaktif. Eksekusi kode JavaScript bergantung pada engine yang ada pada browser. Oleh karena itu, JavaScript disematkan pada kode HTML. Inilah alasan mengapa JavaScript disebut bahasa pemrograman yang bekerja pada sisi client.

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI SISTEM

Pada bab IV ini, akan dijelaskan tentang implementasi atau pengembangan yang telah dilakukan selama pelaksanaan kerja praktik. Pengembangan yang dilakukan adalah sebagai berikut.

4.1 Delivery Time in Shipping Matrix

Sebelum pembeli melakukan *checkout* biasanya calon pembeli terlebih dahulu ingin mengetahui estimasi waktu pengiriman dari barang yang ingin dibeli, maka dari itu perlu ditambahkan estimasi *delivery time* pada detail order. Untuk menambahkan *delivery time* pada *shippingmatrix* maka perlu melakukan pengembangan di beberapa microservice yang berkaitan yakni Morbius, Marvel, dan Marvelui. Berikut pengembangan yang dilakukan di beberapa *microservice* yang terkait.

4.1.1 Marvel UI

Microservices ini bertugas untuk memberikan layanan *interface* kepada *developer* dan *quality assurance* untuk mempermudah dalam melakukan kontrol order dan melakukan testing. Data yang sajikan oleh *microservice* ini merupakan data yang didapat dari *microservices* lain. Pada *microservice* ini penulis membuat tampilan *shipping matrix* yang memuat *delivery time* yang datanya diambil dari *microservice* Marvel. Kode sumber terkait pengembangan pada *microservice* ini dapat dilihat di Kode Sumber 1 pada lampiran.

4.1.2 Marvel

Microservice Marvel adalah *microservice* yang berguna untuk melakukan integrasi order. Saat pengguna melakukan request order maka data – data terkait order tersebut akan diintegrasi menjadi suatu kesatuan detail order sebelum diberikan. Kode sumber terkait

pengembangan pada *microservice* ini dapat dilihat di Kode Sumber 2 dan Kode Sumber 3 pada lampiran.

4.1.3 Morbius

Microservice morbius adalah *microservice* yang berguna untuk melakukan tracking barang dan menghitung estimasi biaya pengiriman suatu order. Untuk melakukan estimasi waktu pengiriman dibutuhkan beberapa data terkait yakni data posisi barang, detail showroom, dan lokasi pembeli. Dari data – data yang telah tersedia tersebut kemudian dilakukan perhitungan sesuai dengan *shipping matrix* yang telah dirancang sebelumnya. Kode sumber terkait pengembangan pada microservice ini dapat dilihat di Kode Sumber 4 dan Kode Sumber 5 pada lampiran.

4.2 *Timeout* pada *Microservice* Marvel

Saat pengguna *microservice* melakukan *request* data detail order ke *microservice* Marvel terkadang terdapat kesalahan atau *bug* pada kode program yang belum diketahui. Hal ini menyebabkan *error* akan tetapi *microservice* tidak menanggapi *request* tersebut sebagai *error* akibatnya pengguna akan menunggu sangat lama dikarenakan tidak terdapat mekanisme *timeout*. Pada kasus ini penulis melakukan pengembangan dengan menambahkan mekanisme *timeout* dalam batas waktu tertentu setelah pengguna melakukan *request* sesuai dengan ketentutan yang diberikan. Kode sumber terkait pengembangan pada *microservice* ini dapat dilihat di Kode Sumber 6 dan Kode Sumber 7 pada lampiran.

4.3 *Refactor* dan Optimasi *Search Order*

Untuk mendapatkan detail order pengembang dapat melakukan *request* detail order ke *microservice marvel* dengan menyertakan *orderid* terkait. Biasanya *request* yang dikirim akan segera mendapatkan *response* namun belakangan ini tidak demikian. Saat ini Fabelio telah memiliki banyak sekali *record* order yang disimpan pada database dan ditambah lagi dengan penggunanya yang terus bertambah sehingga membuat *traffic*

menjadi padat khususnya pada saat mengadakan *campaign* promo. Banyaknya *record* data order dan padatnya *traffic* menyebabkan kecepatan *microservice* menurun dalam merespon *request* yang diterima. Karena adanya permasalahan ini penulis mengatasinya dengan melakukan optimasi dan melakukan *refactor search order* pada *microservice* Marvel. Kode sumber terkait pengembangan pada *microservice* ini dapat dilihat di Kode Sumber 8, 9, dan Kode Sumber 10 pada lampiran.

4.4 *Refactor* Netsuise *Sync* pada Marvel

Microservice Marvel adalah *microservice* yang berguna dalam melakukan integrasi order. Untuk melakukan integrasi order tentunya memerlukan sinkronisasi data salah satunya ERP yang menjadi bagian dari *microservice* Netsuite. Pada pengembangan ini penulis melakukan *refactor* kode untuk melakukan *data sync* setiap lima detik dengan menggunakan cronjob. Kode sumber terkait pengembangan pada *microservice* ini dapat dilihat di Kode Sumber 11 dan Kode Sumber 12 pada lampiran.

4.5 *Execption Handler* dan Inputlog pada Marvel

Pada *microservice* Berapak yang berguna untuk melakukan *handle stock* barang berkemungkinan masih terdapat kendala pada kode sumber yang masih belum diketahui seperti tidak meng-*cover* setiap kasus yang ada. Hal ini dapat menjadi penyebab timbulnya error dikemudian hari dan bisa berakibat fatal terhadap bisnis. Oleh karena itu pada *microservice* ini penulis membuat *exception handler* dan mencatat masalah yang terjadi ke *log* agar permasalahan dapat diketahui dan diperbaiki dengan segera di kemudian hari. Kode sumber terkait pengembangan pada *microservice* ini dapat dilihat di Kode Sumber 13.

4.6 Penerapkan Datadog APM

Saat ini tim NPS memiliki bertanggung jawab terhadap 9 *microservice* yakni marvel, marvelui, payment, PMS, morbius, berapak, netsuite, whatsaap, dan probelio. Dalam melakukan monitoring semua microservice tersebut tentunya menjadi suatu

permasalahan tersendiri. Untuk menyelesaiakan masalah tersebut penulis menggunakan datadog yang merupakan *monitoring service* untuk melakukan *monitoring* setiap *microservice* sehingga *microservice* dapat terawasi dengan mudah. Adapun *microservice* yang ditambahkan datadog pada pengembangan ini yakni *microservice* Whatsapp, Payment, dan Probelio. Kode sumber terkait pengembangan pada *microservice* Whatsapp dapat dilihat di Kode Sumber 14 dan Kode Sumber 15 pada lampiran.

4.7 *Graceful Shutdown* pada *PMS*

Untuk setiap *microservice* pada Fabelio dijalankan menggunakan Docker dengan kontrol menggunakan Kubernetes. Terkadang pada *microservice* khususnya PMS gagal dalam menanggapi *request* ketika *microservice* sedang melayani *request* namun dilakukan *shutdown*. Hal ini akan menyebabkan request tidak mendapatkan *response* maka dari itu hal ini perlu diatasi agar menjamin setiap *request* yang dikirim mendapatkan *response*. Dalam mengatasi ini penulis menerapkan *graceful shutdown* pada Kubernetes sehingga pada saat *microservice* sedang melayani *request* dan dilakukan *shutdown*, *microservice* akan menyelesaikan *request* tersebut terlebih dahulu. Kode sumber terkait pengembangan pada *microservice* ini dapat dilihat di Kode Sumber 16 dan Kode Sumber 17 pada lampiran.

4.8 API Documentation

Dari setiap *endpoint* yang telah dibuat pada setiap *microservice* masih terdapat beberapa *microservice* yang belum terdapat dokumentasi. Hal ini tentunya menjadi hambatan pengembang lain yang perlu menggunakan *endpoint* tersebut. Untuk menghilangkan hambatan tersebut penulis melakukkan dokumentasi untuk setiap *endpoint* pada *microservice* tersebut dan membuat *interface service* pada *microservice* sehingga dapat langsung diakses pengguna. Adapun *microservice* yang didokumentasikan yakni *microservice* Whatsapp dan Morbius. Kode sumber terkait pengembangan pada *microservice* Whatsapp dapat dilihat di Kode Sumber 18 dan Kode Sumber 19 pada lampiran.

[Halaman ini sengaja dikosongkan]

BAB V
PENGUJIAN DAN EVALUASI

Bab ini menjelaskan tahap uji coba dilakukan terhadap pengembangan yang dilakukan pada setiap *microservice*. Pengujian dilakukan untuk memastikan kualitas *microservice* yang dikembangkan dan kesesuaian dengan kebutuhannya. Pengujian dilakukan dengan metode *black box* dimana pengujian yang dilakukan dengan mengamati hasil eksekusi melalui data uji dan memeriksa fungsionalitas dari perangkat lunak dan menggunakan *mocking* apabila diperlukan. Karena perangkat lunak ini merupakan *microservice*, maka pengujian dilakukan dengan menguji fungsionalitas dari setiap fungsi yang dibangun.

5.1 Tujuan Pengujian

Pengujian dilakukan terhadap pengembangan yang telah dilakukan pada *microservice* guna mengetahui beberapa hal berikut ini:

a. Menguji kesesuaian dan ketepatan fungsionalitas dari pengembangan yang dilaukan.
b. Menguji *microservice* agar bisa berjalan tanpa terdapat *error* yang terjadi setelah dilakukan pengembangan.

5.2 Kriteria Pengujian

Penilaian atas pencapaian tujuan pengujian didapatkan dengan memerhatikan beberapa hasil yang diharapkan berikut ini:

a. Kemampuan sistem Marvel untuk memberikan estimasi *delivery time*.
b. Kemampuan sistem untuk memberikan *timeout response*.
c. Kemampuan sistem untuk memberikan respon dengan lebih cepat.
d. Kemampuan sistem untuk melakukan *sync data netsuite*.
e. Kemampuan sistem dalam menangani *exception*.
f. Kemampuan sistem dalam melakakuan *monitoring microservice* dengan Datadog.
g. Kemampuan sistem dalam melakukan *graceful shutdown*.

h. Kemampuan sistem dalam menampilkan dokumentasi terkait.

5.3 Skenario Pengujian

Pengujian dilakukan dengan mencoba mengakses *end point microservice* yang telah dibangun atau mencoba kasus yang menjadi masalah. Diharapkan menghasilkan setiap percobaan menghasilkan output atau perilaku yang sesuai harapan. Skenario pengujian adalah sebagai berikut.

1. User melakukan *request* estimasi order ke Morbius.
2. User melakukan *request* detail order ke Marvel.
3. User mencoba mengakses laman yang disediakan marvelui dan membuka laman *shipping matrix*.
4. User mencoba melakukan *request* berupa *delay* DB dengan menggunakan *mocking*, sistem akan memberhentikan *request* dan merespon *timeout*.
5. User melakukan *request search order* sebelum dan setelah dioptimasi dengan menggunakan bantuan Jmeter untuk membandingkan kecepatan *response*.
6. User mencoba mengupdate salah datu data pada Netsuite kemudian bandingkan data yang telah diupdate dengan data di Marvel setelah lima detik.
7. User mencoba melakukan *request* ke *microservice* yang menyebabkan *error* dengan menggunakan *mocking*.
8. User mencoba mengakses Datadog *board* yang menampilkan detail *microservice* yang sedang dilakukan *monitoring*.
9. User mencoba melakukan *request delay* menggunakan mocking, kemudian *microservice* PMS di-*shutdown*.
10. User mencoba mengakses laman dokumentasi.

5.4 Evaluasi Pengujian

Hasil pengujian dilakukan terhadap pengamatan mengenai perilaku dari setiap pengembangan *microservice* yang telah dilakukan. Setiap pengembangan dievaluasi untuk mengetahui apakah menghasilkan keluaran seperti yang diharapkan

pengembang. Hasil pengujian juga melibatkan *code coverage* yang dihasilkan unit test pada kode sumber program yang telah ditulis.

Berikut ini adalah hasil uji coba terhadap sistem informasi yang telah dibuat :

Tabel 5.1 Tabel Evaluasi Pengujian

Kriteria Pengujian	**Hasil Pengujian**	**Code Coverage**
Microservice morbius memberikan estimasi *delivery time* suatu order.	Terpenuhi	100%
Microservice marvel memberikan detail order dilengkapi dengan *delivery time* suatu order.	Terpenuhi	100%
Microservice marvelui menampilkan shipping matrix dilengkapi dengan *delivery time* suatu order.	Terpenuhi	100%
Microservice marvel memberikan *timeout response*.	Terpenuhi	100%
Microservice marvel memiliki perfomance *speed* yang lebih baik.	Terpenuhi	100%
Microservice Marvel melakukan *sync netsuite* setiap lima detik.	Terpenuhi	100%
Microservice Berapak dapat menangani *exception*.	Terpenuhi	100%

Microsevice termonitoring dan dapat dilihat pada Datadog *board*.	Terpenuhi	100%
Microservice PMS dapat melakukan *graceful shutdown*.	Terpenuhi	100%
Microservice dapat menampillkan dokumentasinya.	Terpenuhi	100%

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Kesimpulan yang didapat setelah melakukan pengembangan di beberapa *microservice* Fabelio adalah sebagai berikut:

- Dengan pengembangan yang telah dilakukan, diharapkan dapat menyelesaikan permasalahan yang menjadi penghambat dalam bisnis.
- Dengan pengembangan yang telah dilakukan, diharapkan dapat membantu proses bisnis Fabelio agar dapat terus berkembang.
- Dari pengembagan yang dilakukan, diharapkan tidak terdapat *bug* yang menyebabkan *error* dikemudian hari.

6.2 Saran

Saran untuk pengembangan *microservice* di Fabelio adalah sebagai berikut:

- Perlunya dibuat dokumentasi dari setiap pengembangan oleh pengembang agar tidak menyebabkan hambatan untuk pengembang lain.
- Perlunya dilakukan analisis lebih lanjut bersama pengembang lain sebelum melakukan pengembangan untuk mengurangi kemungkinan *error* dikemudian hari.

DAFTAR PUSTAKA

1 Shiddiq Ghozali, 2021. Microservices adalah dan Perbedaannya dengan Monolithic Architecture. [Online] Available at: https://datacommcloud.co.id/microservices-adalah-perbedaan-monolithic-architecture [Accessed 25 April 2021].

2 Alexandromeo Lawrence, 2020. *Pengertian API, Fungsi, dan Cara Kerjanya.* [Online] Available at: https://www.niagahoster.co.id/blog/api-adalah/ [Accessed 25 April 2021].

3 Datadog. *Cloud Monitoring as a Service.* [Online] Available at: https://www.datadoghq.com/ [Accessed 25 April 2021].

4 Ahmad Muhardian, 2020. *Pengenalan Dasar MongoDB untuk Pemula.* [Online] Available at: https://www.petanikode.com/tutorial-dasar-mongodb/ [Accessed 25 April 2021].

5 Aldwin Nayoan, 2019. *Pengenalan Node.js Lengkap bagi Pemula.* [Online] Available at: https://www.niagahoster.co.id/blog/node-js-adalah/ [Accessed 25 April 2021].

[Halaman ini sengaja dikosongkan]

LAMPIRAN

```
import _ from 'lodash';
import moment from 'moment';
import React, { Component } from 'react';
import {
 Table,
 TableRow,
 TableCell,
 TableHead,
 TableBody,
} from '@material-ui/core';

import { convertIDR } from './price_tag/convertIDR';

class ShippingAdditionalInfo extends Component {

 render () {
   var item = this.props.item
   const shippingOption = item.shipping_option_detail;
   const stock = shippingOption ? shippingOption.stock
: null;

   return (
     <div style={{ marginTop: '5px' }}
id='itemInfoContainer'>
       <div className='otb-wrapper'>
         {shippingOption && stock ?
           (
             <>
               <Table className="shipping-tbl">
                 <TableHead>
                   <TableRow
className='items_tbl_head'>
                     <TableCell
align='center'>SKU</TableCell>
                     <TableCell align='center'>Is
Kit?</TableCell>
                     <TableCell
align='center'>Shipping Option
Method</TableCell>                    <TableCell
align='center'>Cut-Off Time (COT)</TableCell>
```

```
                        <TableCell align='center'>Out Of
Stock (OOS)</TableCell>
                        <TableCell
align='center'>Expected Delivery Date
(EDD)</TableCell>
                        <TableCell
align='center'>Holidays Found</TableCell>
                        <TableCell
align='center'>Delivery Time</TableCell>
                      </TableRow>
                    </TableHead>
                    <TableBody>
                      <TableRow className='items_tbl'>
                        <TableCell
align='center'>{stock.sku}</TableCell>
                        <TableCell
align='center'>{stock.isKit ? 'Yes' :
'No'}</TableCell>
                        <TableCell
align='center'>{item.shipping_option_method}</TableCel
l>
                        <TableCell
align='center'>{shippingOption.CBM}</TableCell>
                        <TableCell
align='center'>{shippingOption.cutOffTime}</TableCell>
                        <TableCell
align='center'>{shippingOption.outOffStock || '-
'}</TableCell>
                        <TableCell
align='center'>{item.expected_delivery_date &&
moment(item.expected_delivery_date).format('DD MMM
YYYY')}</TableCell>
                        <TableCell
align='center'>{shippingOption.holidaysFound}</TableCe
ll>
                        <TableCell
align='center'>{shippingOption.deliveryTime || '-
'}</TableCell>
                      </TableRow>
                    </TableBody>
                  </Table>
                  <h3>  Interwarehouse</h3>
```

```
                <Table>
                  <TableHead>
                    <TableRow
className='items_tbl_head'>
                      <TableCell
align='center'>Cost</TableCell>
                      <TableCell
align='center'>From</TableCell>
                      <TableCell
align='center'>To</TableCell>
                      <TableCell align='center'>Lead
Time</TableCell>
                    </TableRow>
                  </TableHead>
                  <TableBody>
                    <TableRow className='items_tbl'>
                      <TableCell
align='center'>Rp{convertIDR(shippingOption.interWareh
ouse.cost)},-</TableCell>
                      <TableCell
align='center'>{shippingOption.interWarehouse.from ||
'-'}</TableCell>
                      <TableCell
align='center'>{shippingOption.interWarehouse.to || '-
'}</TableCell>
                      <TableCell
align='center'>{shippingOption.interWarehouse.leadTime
}</TableCell>
                    </TableRow>
                  </TableBody>
                </Table>
                <h3>  Last Mile</h3>
                <Table>
                  <TableHead>
                    <TableRow
className='items_tbl_head'>
                      <TableCell
align='center'>Cost</TableCell>
                      <TableCell align='center'>Lead
Time</TableCell>
                    </TableRow>
                  </TableHead>
```

```
                  <TableBody>
                    <TableRow className='items_tbl'>
                      <TableCell
align='center'>Rp{convertIDR(shippingOption.lastMile.c
ost)},-</TableCell>
                      <TableCell
align='center'>{shippingOption.lastMile.leadTime}</Tab
leCell>
                    </TableRow>
                  </TableBody>
                </Table>
                <h3>  Stock</h3>
                <Table>
                  <TableHead>
                    <TableRow
className='items_tbl_head'>
                      <TableCell
align='center'></TableCell>
                      <TableCell colSpan={3}
align="center">Base Warehouse</TableCell>
                      <TableCell colSpan={3}
align="center">Fulfillment Center</TableCell>
                    </TableRow>
                  </TableHead>
                  <TableBody>
                    <TableRow className='items_tbl'>
                      <TableCell
align='center'>SKU</TableCell>
                      <TableCell
align='center'>Warehouse Name</TableCell>
                      <TableCell align='center'>Qty
Available</TableCell>
                      <TableCell align='center'>Qty
Backordered</TableCell>
                      <TableCell align='center'>FC
Name</TableCell>
                      <TableCell align='center'>Qty
Available</TableCell>
                      <TableCell align='center'>Qty
Backordered</TableCell>
                    </TableRow>
                    {stock.isKit ?
                      stock.kitConfig.map((kit, i) =>
```

```
                        return (
                          <TableRow
className='items_tbl' key={i}>
                            <TableCell
align='center'>{kit.sku}</TableCell>
                            <TableCell
align='center'>{kit.baseWarehouse.warehouseName || '-
'}</TableCell>
                            <TableCell
align='center'>{kit.baseWarehouse.qtyAvailable}</Table
Cell>
                            <TableCell
align='center'>{kit.baseWarehouse.qtyBackOrdered}</Tab
leCell>
                            <TableCell
align='center'>{kit.fulfillmentCenter.warehouseName ||
'-'}</TableCell>
                            <TableCell
align='center'>{kit.fulfillmentCenter.qtyAvailable}</T
ableCell>
                            <TableCell
align='center'>{kit.fulfillmentCenter.qtyBackOrdered}<
/TableCell>
                          </TableRow>
                        )
                        }
                      )
                      :
                      (
                        <>
                          <TableRow
className='items_tbl'>
                            <TableCell
align='center'>{item.sku}</TableCell>
                            <TableCell
align='center'>{stock.baseWarehouse.warehouseName ||
'-'}</TableCell>
                            <TableCell
align='center'>{stock.baseWarehouse.qtyAvailable}</Tab
leCell>
                            <TableCell
align='center'>{stock.baseWarehouse.qtyBackOrdered}</T
ableCell>                     )
```

```
                        </>
                      )
                    }
                  </TableBody>
                </Table>
                {!_.isEmpty(shippingOption.intransitVen
dor) &&
                  <>
                    <h3>  In Transit
Vendor</h3>
                    <Table>
                      <TableHead>
                        <TableRow
className='items_tbl_head'>
                          <TableCell
align='center'>Inbound Date</TableCell>
                          <TableCell
align='center'>Quantity</TableCell>
                          <TableCell
align='center'>Buffer Days</TableCell>
                          <TableCell
align='center'>Buffer Success Rate</TableCell>
                        </TableRow>
                      </TableHead>
                      <TableBody>
                        <TableRow
className='items_tbl'>
                          <TableCell
align='center'>{moment(shippingOption.intransitVendor.
inboundDate).format('DD MMM YYYY')}</TableCell>
                          <TableCell
align='center'>{shippingOption.intransitVendor.qty}</T
ableCell>
                          <TableCell
align='center'>{shippingOption.intransitVendor.bufferD
ays}</TableCell>
                          <TableCell
align='center'>{shippingOption.intransitVendor.bufferS
uccessRate}</TableCell>
                        </TableRow>
                      </TableBody>
                    </Table>
                  </>
```

```
                }
              </>
            )
            :
            (
              <>
                <Table className="shipping-tbl">
                  <TableHead>
                    <TableRow
classNames='items_tbl_head'>
                      <TableCell
align='center'>SKU</TableCell>
                      <TableCell
align='center'>Shipping Option Method</TableCell>
                    </TableRow>
                  </TableHead>
                  <TableBody>
                    <TableRow className='items_tbl'>
                      <TableCell
align='center'>{item.sku}</TableCell>
                      <TableCell
align='center'>{item.is_custom_shipping ? 'Custom
Shipping' : item.shipping_option_method}</TableCell>
                    </TableRow>
                  </TableBody>
                </Table>
              </>
            )
          }
        </div>
        <br/>
        <br/>
      </div>
    )
  }
}

export default ShippingAdditionalInfo
```

Kode Sumber 1 Kode Sumber Delivery TIme MarvelUI

```
const mongooseLeanDefaults = require('mongoose-lean-
defaults');

const mongo = require('../../_helpers/mongoDb');

const { Schema } = mongo;

const shippingOptionSchema = new Schema({
 stock: { type: Object, default: {} },
 interWarehouse: { type: Object, default: {} },
 CBM: { type: Number, default: 0 },
 lastMile: { type: Object, default: {} },
 outOfStock: { type: Number, default: 0 },
 cutOffTime: { type: Number, default: 0 },
 holidaysFound: { type: Number, default: 0 },
 intransitVendor: { type: Object, default: {} },
 deliveryTime: { type: Number, default: 0 },
});
```

Kode Sumber 2 Kode Sumber Order Driver Delivery Time Marvel

```
import { expect } from 'chai';
import axios from 'axios';
import mongoose from 'mongoose';
import SnsPublisher from
'../../../drivers/sns/publisher';

const assert = require('assert');
const sinon = require('sinon');
const fs = require('fs');
const path = require('path');
const config = require('nconf');
const nsreslet = require('nsrestlet');
const NetsuiteRequest =
require('../../../_helpers/netsuiteRequest');
const utils = require('../../../_helpers/utilCls');
const orderModel =
require('../../../drivers/mongo/orders');
const OrderBusiness =
require('../../../src/orders/business');
const FulfillmentCenterLocator =
require('../../../src/orders/fulfillment_center_locato
r');

const netsuiteRequest = new
NetsuiteRequest(config.get('NETSUITE'));
const logger =
require('../../../_helpers/marvelLogger');
describe('Order Business Layer', () => {
 beforeEach(() => {
   sinon
     .stub(FulfillmentCenterLocator.prototype,
'getFulfillmentCenterFromOrder')
     .resolves('10.Karawaci Warehouse : 10.Sell-able
Zone');
   sinon.stub(utils,
'callPmsBySku').resolves('Retail');
 });

 afterEach(() => {
   sinon.restore();
 });
```

```
 describe('sendOrder()', async () => {
   let clock;

   beforeEach(() => {
     clock = sinon.useFakeTimers(new Date('2020-01-01
00:00:00'));
   });

   afterEach(() => {
     clock.restore();
   });

   it('Should return order object with shipping option
additional info for simple item', async () => {
     const payLoad = fs.readFileSync(
       path.resolve(
         __dirname,
         '../../data_samples/createOrderWithShippingOp
tionAdditionalInfo.json',
       ),
       'utf-8',
     );

     const payLoadObject =
JSON.parse(payLoad.toString());

     const sku = payLoadObject.data.item[0].item;
     const shippingOptionDetail = JSON.parse(
       Buffer.from(
         payLoadObject.data.item[0].shipping_option_de
tail,
         'base64',
       ),
     );
     const {
       interWarehouse,
       CBM,
       lastMile,
       outOfStock,
       cutOffTime,
       deliveryTime,
       intransitVendor,
```

```
    } = shippingOptionDetail;

    payLoadObject.store_front_sync = false;
    payLoadObject.erp_sync = false;

    const input = {
      requestName: 'salesorder',
      requestType: 'POST',
      requestParams: '?script=6&deploy=1',
      data: payLoadObject,
    };

    const stubRequest = {
      request: sinon.stub().returns({
        error: false,
        msg: 'A record was successfully created with
Id 1759297',
        id: 1759297,
      }),
    };

    const stubDatabase = {
      find: sinon.stub().returns([]),
      findOne: sinon.stub().returns(null),
      findOneAndUpdate: sinon.stub().returns({
        order_id: input.data.custbody_magento_id,
        netsuite_isSync: true,
        magento_isSync: true,
        items: [
          {
            item_type: 'Retail',
            expected_delivery_date: '2020-08-
26T00:00:00.000Z',
            shipping_option_detail: {
              stock: {
                sku: 'HMDT4S-MMON4SPL',
                isKit: false,
                baseWarehouse: {
                  qtyAvailable: 0,
                  qtyBackOrdered: 1,
                  warehouseName: '10.Karawaci
Warehouse : 10.Sell-able Zone',
```

```
                  },
                },
                interWarehouse: { cost: 20000,
leadTime: 1 },
                CBM: 0.22,
                lastMile: { leadTime: 2, cost: 250000
},
                outOfStock: 30,
                intransitVendor: {
                  inboundDate: '2020-09-18',
                  qty: 100,
                  bufferDays: 14,
                  bufferSuccessRate: 70,
                },
                cutOffTime: 0,
                deliveryTime: 7,
              },
            },
          ],
        }),
      };

      const axiosStub = sinon.stub(axios, 'request');
      axiosStub.onCall(0).resolves({
        data: { product_type: 'Retail' },
      });

      const businessObj = new
OrderBusiness(stubRequest, stubDatabase);
      const response = await
businessObj.sendOrder(input);

      assert.equal(response.status, true);
      assert.equal(response.data.order.magento_isSync,
true);
      assert.equal(response.data.order.netsuite_isSync,
true);
      assert.equal(
        response.data.order.order_id,
        input.data.custbody_magento_id,
      );
```

```
      assert.equal(
        sku,
        response.data.order.items[0].shipping_option_de
tail.stock.sku,
      );
      assert.equal(
        false,
        response.data.order.items[0].shipping_option_de
tail.stock.isKit,
      );
      assert.deepEqual(
        interWarehouse,
        response.data.order.items[0].shipping_option_de
tail.interWarehouse,
      );
      assert.deepEqual(
        CBM,
        response.data.order.items[0].shipping_option_de
tail.CBM,
      );
      assert.deepEqual(
        lastMile,
        response.data.order.items[0].shipping_option_de
tail.lastMile,
      );
      assert.deepEqual(
        outOfStock,
        response.data.order.items[0].shipping_option_de
tail.outOfStock,
      );
      assert.deepEqual(
        cutOffTime,
        response.data.order.items[0].shipping_option_de
tail.cutOffTime,
      );
      assert.deepEqual(
        deliveryTime,
        response.data.order.items[0].shipping_option_de
tail.deliveryTime,
      );
```

```
      assert.deepEqual(
        intransitVendor,
        response.data.order.items[0].shipping_option_de
tail.intransitVendor,
      );
    });
  });
});
```

Kode Sumber 3 Kode Sumber Unit Test Delivery Time Marvel

```
import _ from 'lodash';
import { logFormat } from '../../_helpers/utils';
import utils from './utils';
import {
 FULFILLMENT_BY_FABELIO,
 MARKETPLACE,
} from './definition';

export default class ItemShippingBiz {
 static getOptions(params) {
   const {
     sku,
     parentProduct,
     childSku,
     qtyParent,
     zipcode,
     productsStock,
     productInfo,
     productBaseWarehouses,
     fulfillmentCenter,
     shippingTypes,
     lastMiles,
     lastMileDefault,
     interWarehouses,
     interWarehouseDefault,
     cutOffTime,
     holidays,
     startFrom,
     inTransitVendors,
   } = params;
   let logInfos = [];
   const skuQuantity = _.toNumber(sku.quantity);
   const skuCode = sku.sku;
   const formatInput = { sku: skuCode, zipcode };
   const {
     product_type: productType,
     attributes: productAttributes,
   } = productInfo;

   const [productStock] = productsStock.filter(product
=> product.sku === skuCode);
   if (_.isEmpty(productStock) && ![MARKETPLACE,
FULFILLMENT_BY_FABELIO].includes(productType)) {
```

```
      logInfos.push(logFormat({ sku: skuCode, zipcode
}, 'product-stock-notfound'));
    }

    const { value: productBaseWarehouse, log:
logProductBW } = utils.getProductBaseWarehouse(
      formatInput, productBaseWarehouses,
    );
    if (logFC) logInfos.push(logOOSLT);

    const FCAvailableStock = _.subtract(FCStock,
skuQuantity);
    const availableShippingOptions = [];

    shippingTypes.map((shippingType) => {
      const displayName = shippingType.display_name;
      const cutOff = (shippingType.exclude_cutoff) ? 0
: cutOffTime;

      if (shippingType.is_strict) {
        if
(_.isEmpty(shippingType.product_shipping_types)) {
          logInfos.push(logFormat({
            input: formatInput, shipping_type:
displayName,
          }, 'product-strict-shipping-type-empty'));
          return false;
        }

        const productShippingType =
shippingType.product_shipping_types.filter(productST
=> (
          _.isEqual(productST.sku, skuCode)
        ));
        if (_.isEmpty(productShippingType)) {
          logInfos.push(logFormat({
            input: formatInput, shipping_type:
displayName,
          }, 'product-strict-shipping-type-notfound'));
          return false;
        }
      }
```

```
      const { value: lastMileCostTime, log:
logLMCostTime } = utils.getLastMileCostTime(
        {
          input: formatInput,
          shipping_type: displayName,
          lastMiles,
          lastMileDefault,
          shippingTypeId: shippingType.id,
          isStrict: shippingType.is_strict,
        },
      );
      if (logLMCostTime) logInfos.push(logLMCostTime);

      if (!lastMileCostTime) return false;

      const {
        cost: lastMileCost,
        leadTime: lastMileLT,
      } = lastMileCostTime;

      let interWarehouseCost = 0;
      let interWarehouseLT = 0;
      let BWAvailableStock = FCStock;
      if (
        FCAvailableStock < 0
        && !_.isEqual(fulfillmentCenter,
productBaseWarehouse)
        && !_.isEqual(productType, 'Marketplace')
      ) {
        logInfos.push(logFormat({
          formatInput, shipping_type: displayName,
        }, 'accept-inter-warehouse'));

        const { value: IWCostTime, log: logIWCostTime }
= utils.getInterWarehouseCostTime(
          {
            input: formatInput,
            shipping_type: displayName,
            interWarehouses,
            interWarehouseDefault,
            productBaseWarehouse,
            fulfillmentCenter,
          },
```

```
        );
      let productDeliveryTime;
      if (
        (productStock && productType ===
FULFILLMENT_BY_FABELIO)
        || (productStock && productStock.productType
=== FULFILLMENT_BY_FABELIO)
      ) {
        BWAvailableStock = productStock.locations
          .reduce((prev, curr) => prev +
+curr.qtyAvailable, 0);

        productDeliveryTime =
+productInfo.attributes.deliverytime;
      }

      const totalAvailableStock =
_.subtract(BWAvailableStock, skuQuantity);
      const {
        value: leadTimeWithoutHoliday,
        log: logLTWithoutHoliday,
      } = utils.getLeadTimeWithoutHoliday({
        input: formatInput,
        shipping_type: displayName,
        productType,
        lastMileLT,
        interWarehouseLT,
        totalAvailableStock,
        productLTBackOrder,
        productDeliveryTime,
        startFrom,
        cutOffTime: cutOff,
      });
      if (logLTWithoutHoliday)
logInfos.push(logLTWithoutHoliday);

      const {
        value: leadTimeWithHoliday,
        log: logLTWithHoliday,
        holidaysFound,
      } = utils.getEstimatedWithHoliday({
        input: formatInput,
        shippingType: displayName
```

```
        productType,
        leadTimeWithoutHoliday,
        holidays,
        isStockAvailable: (totalAvailableStock >= 0),
        baseWarehouse: productBaseWarehouse,
        fulfillmentCenter,
        startFrom,
      });
      if (logLTWithHoliday.length > 0) logInfos =
[...logInfos, ...logLTWithHoliday];

      const skuFinalQty = qtyParent || skuQuantity;
      const shippingCost = utils.getShippingCost(
        lastMileCost, interWarehouseCost, productCBM,
skuFinalQty,
      );

      let usedInterWarehouse = {
        cost: interWarehouseCost,
        leadTime: interWarehouseLT,
        from: null,
        to: null,
      };

        usedInterWarehouse = { ...usedIWCost };
      }

      const stock = () => {
        const validProductStock = !parentProduct ?
productStock : parentProduct;
        const validWarehouseName = !parentProduct ?
productBaseWarehouse : undefined;
        const validFC = !parentProduct ?
fulfillmentCenter : undefined;

        const {
          value: qtyAvailableFC,
          qtyBackOrdered: qtyBackOrderedFC,
        } = utils.getStockByWarehouse(
          formatInput,
          validProductStock,
          fulfillmentCenter,
        );
```

```
        const payload = {};
        payload.sku = skuCode;
        payload.isKit = !_.isEmpty(parentProduct);
        payload.baseWarehouse = BWSimple;
        payload.fulfillmentCenter = FCSimple;

        if (parentProduct) {
          payload.sku = parentProduct.sku;

          payload.kitConfig =
productsStock.filter(child =>
childSku.includes(child.sku))
            .map((child) => {
              const {
                value: childQtyAvailable,
                qtyBackOrdered: childQtyBackOrdered,
              } = utils.getStockByWarehouse(
                formatInput,
                child,
                productBaseWarehouse,
              );

              return {
                sku: child.sku,
                baseWarehouse: {
                  qtyAvailable: childQtyAvailable,
                  qtyBackOrdered: childQtyBackOrdered,
                  warehouseName: productBaseWarehouse,
                },
                fulfillmentCenter: {
                  qtyAvailable: childQtyAvailableFC,
                  qtyBackOrdered:
childQtyBackOrderedFC,
                  warehouseName: fulfillmentCenter,
                },
              };
            });
        }

        return payload;
      };
```

```
      availableShippingOptions.push({
        code: shippingType.shipping_type,
        name: shippingType.display_name,
        estimatedDeliveryDays:
leadTimeWithoutHoliday.leadTime,
        estimatedDeliveryDate:
leadTimeWithHoliday.expectedDate,
        estimatedDeliveryDaysWithHoliday:
leadTimeWithHoliday.leadTime,
        cost: shippingCost,
        isOutOfStock: (totalAvailableStock < 0),
        detail: {
          stock: stock(),
          interWarehouse: usedInterWarehouse,
          CBM: productCBM,
          lastMile: {
            leadTime: lastMileLT,
            cost: lastMileCost,
          },
          outOfStock: productLTBackOrder,
          intransitVendor: productLTInfo,
          cutOffTime,
          holidaysFound,
          deliveryTime: productInfo.attributes &&
+productInfo.attributes.deliverytime,
        },
      });

      return true;
    });

    return {
      value: _.orderBy(availableShippingOptions,
'estimatedDeliveryDays', 'asc'),
      log: logInfos,
    };
  }
```

Kode Sumber 4 Kode Sumber Use-case Delivery Time Morbius

```
import sinon from 'sinon';
import { expect } from 'chai';
import request from 'supertest';
import sequelize from 'sequelize';
import axios from 'axios';
import Redis from 'ioredis';

import logger from
'../../../../src/_helpers/morbiusLogger';
import app from '../../../../src/interfaces/http';

describe('POST /shipping-matrix', () => {
 const dateNow = new Date('September 27, 2019
07:00:00');
 let sandbox;

 process.env.INCLUDE_IN_TRANSIT = true;

 beforeEach(() => {
   sandbox = sinon.createSandbox();
 });

 afterEach(() => {
   sandbox.restore();
 });

 it('Should return 200 with listing available shipping
methods with estimated delivery, cost, and detail',
async () => {
   const input = {
     skus: [
       {
         sku: 'FMPDLCCT002-WH',
         quantity: 3,
       },
     ],
     zipcode: '11630',
   };

   const stubClock = sandbox.useFakeTimers(dateNow);
   const stubFindAll = sandbox.stub(sequelize.Model,
'findAll');
```

```
    const stubAxios = sandbox.stub(axios, 'post');
    const stubFindOne = sandbox.stub(sequelize.Model,
'findOne');
    const stubCount = sandbox.stub(sequelize.Model,
'count');
    const stubRedisGet = sandbox.stub(Redis.prototype,
'get');
    const stubRedisSet = sandbox.stub(Redis.prototype,
'set');
    const stubRedisPipeline =
sandbox.stub(Redis.prototype, 'pipeline');

    // stub pms
    stubRedisPipeline.onCall(0).returns({
      get: sandbox.stub().returns([[null, null]]),
      exec: sandbox.stub().resolves([[null, null]]),
    });
    stubRedisSet.onCall(0).resolves(true);
    stubAxios.onCall(0).resolves({
      data: [
        {
          sku: input.skus[0].sku,
          product_type: 'Retail',
          attributes: {
            product_volume: '3',
            deliverytime_backorder: '5 days',
            deliverytime: '7',
          },
        },
      ],
    });

    // stub shipping types
    stubRedisGet.onCall(0).resolves(null);
    stubRedisSet.onCall(1).resolves(true);
    stubFindAll.onCall(0).returns([
      {
        id: 1,
        shipping_type: 'standard',
        display_name: 'Standard',
        is_strict: false,
        product_shipping_types: [],
      },
```

```
    // stub inter warehouses
    stubRedisPipeline.onCall(2).returns({
      get: sinon.stub().returns([[null, null]]),
      exec: sinon.stub().resolves([[null, null]]),
    });
    stubRedisSet.onCall(4).resolves(true);
    stubRedisGet.onCall(2).resolves(null);
    stubRedisSet.onCall(5).resolves(true);
    stubFindOne.onCall(2).returns({ unit_cost: 5000,
lead_time: 3 });
    stubCount.onCall(0).returns(1);
    stubFindAll.onCall(1).returns([
      {
        warehouseFrom: { display_name: '10.Karawaci
Warehouse' },
        warehouseTo: { display_name: '07.Bandung
Warehouse' },
        unit_cost: 0,
        lead_time: 3,
      },
    ]);

    // stub location holiday
    stubRedisGet.onCall(5).resolves(null);
    stubFindOne.onCall(6).returns({ id: 1 });
    stubFindAll.onCall(3).returns([
      {
        holiday_date: '2019-07-17',
        warehouse: { id: 1, display_name: '10.Karawaci
Warehouse : 10.Sell-able Zone' },
      },
    ]);
    stubRedisSet.onCall(8).resolves(true);

    stubRedisGet.onCall(6).resolves(null);
    stubFindOne.onCall(7).returns({ id: 1 });
    stubFindAll.onCall(4).returns([
      {
        holiday_date: '2019-07-18',
        warehouse: { id: 2, display_name: '07.Bandung
Warehouse : 07.Sell-able Zone' },
```

```
    // stub in transit vendor
    stubRedisPipeline.onCall(3).returns({
      get: sinon.stub().returns([[null, null]]),
      exec: sinon.stub().resolves([[null, null]]),
    });
    stubFindAll.onCall(5).returns([]);

    const { status, body } = await request(app)
      .post('/rest/v1/shipping-matrix')
      .send(input);

    stubClock.restore();

    expect(status).to.equal(200);
    expect(body.status);
    expect(body.data).to.be.an('array');
    expect(body.data).to.have.length(1);
    expect(body.data[0].shippingOptions).to.have.length
(1);
    expect(body.data[0].shippingOptions[0].detail.stock
.sku).to.equal('FMPDLCCT002-WH');
    expect(body.data[0].shippingOptions[0].detail.stock
.isKit).to.equal(false);
    expect(body.data[0].shippingOptions[0].detail.inter
Warehouse.leadTime).to.equal(0);
    expect(body.data[0].shippingOptions[0].detail.CBM).
to.equal(3);
    expect(body.data[0].shippingOptions[0].detail.outOf
Stock).to.equal(5);
    expect(body.data[0].shippingOptions[0].detail.lastM
ile.cost).to.equal(5000);
    expect(body.data[0].shippingOptions[0].detail.lastM
ile.leadTime).to.equal(3);
    expect(body.data[0].shippingOptions[0].detail.holid
aysFound).to.equal(0);
    expect(body.data[0].shippingOptions[0].detail.deliv
eryTime).to.equal(7);
    expect(body.data[0].shippingOptions[0].estimatedDel
iveryDays).to.equal(3);
    expect(body.data[0].shippingOptions[0].isOutOfStock
).to.equal(false);
    expect(body.data[0].shippingOptions[0].cost).to.equ
al(45000);
```

Kode Sumber 5 Kode Sumber Unit Test Delivery Time Morbius

```
const mongoose = require('mongoose');
const logger = require('./marvelLogger');
const nconf = require('../config');

const { server, port, database, replica } =
nconf.get('DATABASE');
const user = nconf.get('DATABASE_USER');
const password = nconf.get('DATABASE_PASSWORD');
const socketTimeout = nconf.get('SOCKET_TIMEOUT');
const queryTimeout = nconf.get('QUERY_TIMEOUT');

const replicaUrl = !replica ? '' :
`&replicaSet=${replica}`;
const mongoUrl =
`mongodb://${server}:${port}/${database}?authSource=ad
min${replicaUrl}`;
const connectionObj = {
 useNewUrlParser: true,
 useUnifiedTopology: true,
 socketTimeoutMS: socketTimeout,
};

if (user && password) connectionObj.auth = { user,
password };

if (nconf.get('NODE_ENV') !== 'test') {
 mongoose.connect(mongoUrl,
connectionObj).catch((error) => {
   logger.error(error, 'Initial-mongo-connect-error');
   process.exit(1);
 });
 mongoose.set('useCreateIndex', true);
 mongoose.set('bufferCommands', false);
 mongoose.set('maxTimeMS', queryTimeout);
}

mongoose.set('useFindAndModify', false);

module.exports = mongoose;
```

Kode Sumber 6 Kode Sumber Helper MongoDB Timeout Marvel

```
import { expect } from 'chai';
import sinon from 'sinon';
const mongoose = require('../../_helpers/mongoDb');

describe('MangoDB Timeout Test', () => {
 // eslint-disable-next-line func-names
 it('Timeout Test', async function () {
   this.timeout(100000);
   const stubFind = sinon.stub(mongoose.Model,
'find');
   stubFind.onCall(0).rejects({ name: 'TypeError' });
   stubFind.onCall(1).rejects({ name:
'MongoNetworkError' });
   stubFind.onCall(2).rejects({ message: 'operation
exceeded time limit' });

   const userModel = mongoose.model('users');

   let connectionError;
   try {
     await userModel.find({});
   } catch (error) {
     connectionError = error.name;
   }

   let socketError;
   try {
     await userModel.find({ $where: 'sleep(90000) ||
true' });
   } catch (error) {
     socketError = error.name;
   }

   let maxQueryError;
   try {
     await userModel.find({ $where: 'sleep(60000) ||
true' });
   } catch (error) {
     maxQueryError = error.message;
   }
```

```
    expect(connectionError).to.equal('TypeError');
    expect(socketError).to.equal('MongoNetworkError');
    expect(maxQueryError).to.equal('operation exceeded
time limit');
  });
});
```

Kode Sumber 7 Kode Sumber Unit Test Timeout Marvel

```
const mongooseLeanDefaults = require('mongoose-lean-
defaults');
const mongo = require('../../_helpers/mongoDb');
const { Schema } = mongo;

const ordersSchema = new Schema({
 increment_id: { type: String, default: null },
 order_id: { type: Number, default: 0 },
 customer_id: { type: Number, default: 0 },
 customer_email: { type: String, default: null },
 customer_firstName: { type: String, default: null },
 customer_lastName: { type: String, default: null },
 customer_phone: { type: String, default: null },
 address: { type: Array },
 source: { type: String, default: null },
 channel: { type: String, default: null },
 status: { type: String, default: null },
 netsuite_status: { type: String, default: null },
 magento_status: { type: String, default: null },
 netsuite_isSync: { type: Boolean, default: false },
 magento_isSync: { type: Boolean, default: false },
 agent: { type: String, default: null },
 showroom: { type: String, default: null },
 discount: { type: Number, default: 0 },
 total: { type: Number, default: 0 },
 discount_descriptions: { type: String },
 location: { type: String, default: null },
 items: [itemsSchema],
 netsuite_sync: { type: Number, default: 0 },
 shipping_total: { type: Number, default: 0 },
 payment_method: { type: String, default: null },
 payment_method_title: { type: String, default: null
},
 payment_detail: paymentSchema,
 created_at: { type: Date },
 modified_at: { type: Date, default: Date.now },
 delivery_type: { type: String, default: null },
 delivery_slot: { type: String, default: null },
 installation_slot: { type: String, default: null },
 is_sync: { type: Boolean, default: false },
 rdd_days: { type: Number, default: 0 },
 rid_days: { type: Number, default: 0 },
 is_emit: { type: Boolean, default: false },
```

```
 is_onedaydelivery: { type: Boolean, default: false },
 is_discounteditem: { type: Boolean, default: false },
 magento_processing_at: { type: Date, default: null },
 notification_processing_time: { type: Date, default:
null },
 transfer_reminder_sent: { type: Date, default: null
},
 grouped_discount: [groupedDiscountSchema],
});

ordersSchema.index({ created_at: -1 });
ordersSchema.index({ increment_id: -1 });
ordersSchema.index({ customer_phone: 1 });
ordersSchema.index({ customer_email: 1 });
ordersSchema.index({
 customer_firstname: 'text',
 customer_lastname: 'text',
});
```

Kode Sumber 8 Kode Sumber Driver Mongo Search Order Marvel

```
import _ from 'lodash';
import moment from 'moment';
import express from 'express';
import uuid from 'uuid';

import orders from
'../../../migrations/exportOrdersWithoutRdd';
import RidBiz from '../../../src/rid/business';
import installationPlanModel from
'../../../drivers/mongo/installation_plans';
import holidaysDriver from
'../../../drivers/morbius/holidays';
import InstallationPlanBiz from
'../../../src/installationPlan/business';

const logger =
require('../../../_helpers/marvelLogger');

const validator =
require('./validation/customerOrder');
const validate =
require('../../../_helpers/expressValidator');
const to = require('../../../_helpers/to');
const utils = require('../../../_helpers/utils');
const commonResponse =
require('../../../_helpers/commonResponse');

const nconf = require('../../../config');
const netsuite =
require('../../../_helpers/netsuiteRequest')(
 nconf.get('NETSUITE'),
);
const orderModel =
require('../../../drivers/mongo/orders');
const confirmModel =
require('../../../drivers/mongo/confirmation_tracking'
);
const historyModel =
require('../../../drivers/mongo/order_history');
const SendGrid =
require('../../../drivers/email/sendgrid');
const DeliveryPlanBiz =
require('../../../src/deliveryPlan/business');
```

```
const orderHistoryDao = new
OrderHistoryDao(historyModel);
const orderDao = new OrderDao(orderModel);
const orderBiz = new OrderBiz(
 netsuite,
 orderModel,
 producer,
 rddBiz,
 historyBiz,
 {},
);
const orderSearchBiz = new OrderSearchBiz(
 netsuite,
 orderModel,
 producer,
 rddBiz,
 historyBiz,
 {},
);
const installationPlanBiz = new InstallationPlanBiz(
 installationPlanModel,
 orderBiz,
 {},
 holidaysDriver,
);

const topic = nconf.get('SQS:orderSetting:QueueUrl');
module.exports = () => {
 const router = express.Router();

router
    .route('/order_id/:orderId')
    .get(validate(validator.getOrderByOrderId), async
(req, res) => {
      const tid = uuid.v4();

      const { data } = await
to(orderSearchBiz.getOrderDetailsbyOrderId(req, tid));

      commonResponse.sendResponse(data, res);
    });
};
```

Kode Sumber 9 Kode Sumber Interface Serach Order Marvel

```
import CanRidBusiness from '../rid/business_canRid';
import MAP_SLOT from '../definitions/mapSlot';

const _ = require('lodash');
const mongoose = require('mongoose');
const uuid = require('uuid');

const utils = require('../../_helpers/utils');
const orderItemService =
require('../../services/orderItemService');
const UtilCls = require('../../_helpers/utilCls');
const OrderDao = require('./dao');
const FulfillmentCenterLocator =
require('./fulfillment_center_locator');
const CanRddBusiness =
require('../rdd/business_canRdd');
const { HISTORY_LABELS } =
require('../history/constants');

const orderSearchBusiness = () => {
 class OrderSearchBusiness {
   constructor(
     orderSource,
     dbDriver,
     queueDriver,
     emailSource,
     historySource,
     addressHistorySource,
     ridSource,
   ) {
     this.orderSource = orderSource;
     this.dao = new OrderDao(dbDriver);
     this.queueDriver = queueDriver;
     this.emailSource = emailSource;
     this.historySource = historySource;
     this.orderItemService = orderItemService;
     this.addressHistorySource = addressHistorySource;
     this.ridSource = ridSource;
     this.tid = uuid.v4();
     this.fulfillmentCenterLocator = new
FulfillmentCenterLocator();
     this.util = UtilCls;
     this.canRddBusiness = new CanRddBusiness();
```

```
      this.canRidBusiness = new CanRidBusiness();
      this.mapSlot = MAP_SLOT;
    }

    async aggregateOrderDetails(params) {
      const { filterCriteria, sortCriteria,
offsetCriteria } = params;
      let result = {};

      result = {
        orders: await this.dao.getOrderDetails(
          filterCriteria,
          sortCriteria,
          offsetCriteria,
        ),
        totalCount: await
this.dao.getOrderCount(filterCriteria),
      };

      return result;
    }
    async getOrderDetails(params) {
      const { orders, totalCount } = await
this.aggregateOrderDetails(params);

      const { offsetCriteria, withErrors, tid } =
params;
      const returnData = {};

      if (_.isEmpty(orders)) {
        returnData.data = { orders: [] };
        returnData.status = false;
        returnData.message = 'No such Order, please
check again';

        return returnData;
      }

      const ifErrorPromiseContainer = [];
      orders.forEach((order) => {
        const { items, increment_id: incrementId } =
order;
```

```
        if (withErrors) {
          ifErrorPromiseContainer.push(
            this.historySource.fetchNetsuiteHistory({
              filterCriteria: {
                order_id: incrementId,
                status: HISTORY_LABELS.SKU_MISMATCH,
              },
              tid,
            }),
          );
        }

        _.update(order, 'delivery_slot', (n) =>
this.util.mapSlotName(n));
        _.update(order, 'installation_slot', (n) =>
this.util.mapSlotName(n));

        items.forEach((item, index) => {
          const itemRdd =
utils.validDate(item.requested_delivery_date);
          const itemRid =
utils.validDate(item.requested_installation_date);

          items[index] = _.assign(
            {},
            item instanceof mongoose.Document ?
item.toObject() : item,
            {
              requested_delivery_date: itemRdd,
              can_update_rdd:
this.canRddBusiness.canUpdateRdd(order, item),
              requested_installation_date: itemRid,
              can_update_rid:
this.canRidBusiness.canUpdateRid(order, item),
            },
          );
        });
      });

      if (withErrors) {
        const errorRecords = await
Promise.all(ifErrorPromiseContainer);
```

```
        if (errorRecords.length > 0) {
          errorRecords.forEach((errorDetails,
errorIndex) => {
            const error = errorDetails.status ?
errorDetails.data : [];
            const orderDetails =
_.cloneDeep(orders[errorIndex]);
            const currentOrderDetails =
              orderDetails instanceof mongoose.Document
                ? orderDetails.toObject()
                : orderDetails;
            orders[errorIndex] = _.assign({},
currentOrderDetails, { error });
          });
        }
      }

      returnData.status = true;
      returnData.message = 'OK';
      returnData.data = {
        orders,
        current_page: offsetCriteria.page,
        total_page: _.ceil(totalCount /
offsetCriteria.limit),
        total_records: totalCount,
      };

      return returnData;
    }

    async getOrderDetailsbyOrderId(req, tid) {
      const { orderId } = req.params;
      const { withErrors } = req.query;
      let { page, limit } = req.query;

      page = page || 1;
      limit = limit || 10;
      const params = {
        filterCriteria: { increment_id: orderId },
        offsetCriteria: {
          start: (page - 1) * limit,
          page,
          limit,
```

```
        },
        sortCriteria: { created_at: -1 },
        withErrors,
        tid,
      };

      return this.getOrderDetails(params);
    }
  }

  return OrderSearchBusiness;
};

module.exports = orderSearchBusiness();
```

Kode Sumber 10 Kode Sumber Business Serach Order Marvel

```
import uuid from 'uuid';

const cron = require('node-cron');
const to = require('../../_helpers/to');
const logger = require('../../_helpers/marvelLogger');
const nconf = require('../../config');
const netsuite =
require('../../_helpers/netsuiteRequest')(
 nconf.get('NETSUITE'),
);
const stoStarter = require('./stoStarter');

const orderModel =
require('../../drivers/mongo/orders');
const historyModel =
require('../../drivers/mongo/order_history');
const confirmModel =
require('../../drivers/mongo/confirmation_tracking');

const OrderBiz = require('../../src/orders/business');
const SyncKitBiz =
require('../../src/orders/business_synckit');
const SyncBiz =
require('../../src/orders/business_sync');
const RddBiz = require('../../src/rdd/business');
const HistoryBiz =
require('../../src/history/business');

const sendGridCredentials = nconf.get('SENDGRID');
sendGridCredentials.API_KEY =
nconf.get('SENDGRID_API_KEY');

const historyBiz = new HistoryBiz(historyModel);
const rddBiz = new RddBiz(nconf, confirmModel,
sendGrid, historyBiz);
const orderBiz = new OrderBiz(
 netsuite,
 orderModel,
 producer,
 rddBiz,
 historyBiz,
);
```

```
const syncBiz = new SyncBiz(netsuite, orderBiz);
const syncKitBiz = new SyncKitBiz(netsuite, orderBiz);

const fetchNetsuite = cron.schedule(
 '*/5 * * * * *',
 async () => {
   const tid = uuid.v4();

   logger.info(
     'Starting fetch netsuite line items service',
     'fetch-netsuite-line-items-input',
     tid,
   );

   const { error } = await
to(syncBiz.callRequest(tid));
   if (error) {
     logger.error(error, 'Netsuite-sync-exception');
   }
   /**
    * Sync kit data of orders
    */
   await syncKitBiz.callRequest(tid);
   /**
    * Call Sto
    */
   await stoStarter(tid);
 },
 { scheduled: false },
);

if (process.env.NODE_ENV !== 'test') {
 fetchNetsuite.start();
}

module.exports = fetchNetsuite;
```

Kode Sumber 11 Kode Sumber Interface Sync Netsuite Marvel

```
import cron from 'node-cron';
import uuid from 'uuid';
import sinon from 'sinon';
import mongoose from 'mongoose';
import { expect } from 'chai';

import logger from '../../../_helpers/marvelLogger';
import fetchNetsuite from
'../../../interfaces/services/fetchNetsuiteLineItems';
import SyncBiz from
'../../../src/orders/business_sync';

const nsrestlet = require('nsrestlet');

describe('fetch netsuite line items test', () => {
 let clock;

 beforeEach(() => {
   clock = sinon.useFakeTimers();
 });

 afterEach(() => {
   sinon.restore();
   clock.restore();
   fetchNetsuite.stop();
 });

 it('should validate a uuid as generated in cronjob',
async () => {
   let tid;

   cron.schedule('* * * * * *', () => {
     tid = uuid.v4();
   });
   clock.tick(2000);

   expect(tid).match(v4);
 });
```

```
 it('should processed scheduled task', () => {
   const netsuiteStub = sinon.stub(nsrestlet,
'createLink');
   const accountSettings = {
     accountId: '4819062_SB1',
     tokenKey: 'netsuite_token_public',
     tokenSecret: 'netsuite_token_secret',
     consumerKey: 'netsuite_consumer_public',
     consumerSecret: 'netsuite_consumer_secret',
   };
   const urlSettings = {
     url: 'https://dummy.com?script=70&deploy=1',
     contentType: 'application/json',
   };

   let res = JSON.stringify([
     {
       id: '654258',
       recordtype: 'serializedinventoryitem',
       columns: {
         itemid: '518928-01-01-2725-01',
         baseprice: 199500,
         quantity: 1,
         quantitycommitted: 1,
         tranid: 'ORD-20-07-10-721779250',
         line: 5,
       },
     },
   ]);
   netsuiteStub.withArgs(accountSettings,
urlSettings).returns({
     get: sinon.stub().resolves(res),
   });

   res = JSON.stringify({
     pagination: {
       per_page: '1',
       count: 87,
       page: '1',
       total_page: 1,
     },
     data: [
```

```
        {
          values: {
            'GROUP(trandate)': '2020-11-7',
            'GROUP(tranid)': 'ORD-20-11-07-490076920',
            'GROUP(statusref)': [{ value: 'SalesOrd:G',
text: 'Billed' }],
            'GROUP(customer.entityid)': '80550',
            'GROUP(formulatext)': '+6281228881000',
            'GROUP(custcol_fab_exp_dlvry_time)': '2020-
11-07 01:09:56',
            'GROUP(item)': [
              { value: '47184', text: 'F04FD25-3KH0000-
0 : F04FD25-3KH00000' },
            ],
            'MAX(lineSystemNotes.date)': '2020-11-16
6:53 pm',
            'MAX(linelastmodifieddate)': '2020-11-16
6:54 pm',
            'GROUP(custbody_delivery_slot)': [{ value:
'', text: '- None -' }],
            'GROUP(custbody_installation_slot)': [
              { value: '', text: '- None -' },
            ],
          },
        },
      ],
    });
    netsuiteStub.returns({
      get: sinon.stub().resolves(res),
    });

    sinon.stub(mongoose.Model, 'find').resolves([
      {
        increment_id: 'ORD-345',
        items: [345],
        event_type: 'order-shipped',
        sent_at: '2020-02-06T11:16:00.000Z',
      },
    ]);
    sinon.stub(mongoose.Model,
'findOneAndUpdate').returns(true);
```

```
    sinon.stub(mongoose.Model,
'updateMany').returns(true);
    sinon.stub(mongoose.Model, 'aggregate').returns();
    const spyLogger = sinon.spy(logger, 'info');
    const spySyncBiz = sinon.spy(SyncBiz.prototype,
'callRequest');

    fetchNetsuite.start();
    expect(fetchNetsuite.getStatus()).to.equal('schedul
ed');

    clock.tick(18000);
    expect(fetchNetsuite.getStatus()).to.equal('running
');

    expect(spyLogger.callCount).to.equal(3);
    expect(spyLogger.args[0][0]).to.equal(
      'Starting fetch netsuite line items service',
    );
    expect(spyLogger.args[0][2]).match(v4);
    expect(spySyncBiz.callCount).to.equal(3);
    sinon.assert.calledThrice(netsuiteStub);
  });
});
```

Kode Sumber 12 Kode Sumber Unit Test Sync Netsuite Marvel

```
import uuidv4 from "uuid/v4";
import { RateLimiterMemory, RateLimiterRedis } from
"rate-limiter-flexible";

import { Logger } from "../../../helpers/logger";
import { container } from "../../../container";

const logger = new Logger();

const GetStockUseCase: GetStockUseCaseInterface =
container.resolve(
 "getStockUseCase"
);
const StockWebhookUseCase:
StockWebhookUseCaseInterface = container.resolve(
 'StockWebhookUseCase'
);
const GetInTransitUseCase:
GetInTransitUseCaseInterface = container.resolve(
 "getInTransitUseCase"
);
const rateLimiter: RateLimiterMemory |
RateLimiterRedis = container.resolve(
 "rateLimiter"
);

const corsHeaders = {
 'access-control-allow-origin': '*',
 'access-control-allow-headers': 'Origin, X-Requested-
With, Content-Type, Accept, X-PINGOTHER',
 'access-control-allow-methods': 'GET, HEAD, POST,
OPTIONS'
};

module.exports = function(fastify, opts, next) {
 fastify.options('*', (req, reply) => {
   reply
     .code(204)
     .headers(corsHeaders)
     .send();
 });
```

```
 fastify.get(
   "stockByLocation/:locationId",
   {
     schema: {
       description: "return stock details based on
location",
       tags: ["berapak"],
       summary: "Stock based on location",
       params: {
         type: "object",
         properties: {
           locationId: {
             type: "string",
             description: "Location ID and name",
           },
         },
       },
       query: {
         type: "object",
         properties: {
           page: { type: 'number' },
           limit: { type : 'number'},
         }
       },
       required: ['locationId'],
     },
   },
   async (req, reply) => {
     const tid = uuidv4();
     const locationId =
decodeURI(req.params.locationId);
     const page = Number(req.query.page) || 1;
     const limit = Number(req.query.limit < 100 ?
req.query.limit : 100) || 100;

     logger.info(req.params, "stockByLocation: input",
tid);

     if(!locationId) {
       const result = { status: false, message:
"Invalid Input" };
```

```
        logger.error(result, "stockByLocation: output",
tid);
        reply.send(result);
      }

      reply.headers(corsHeaders);

      const result = await
GetStockUseCase.getStockRelatedDetailsByLocation(locat
ionId, page, limit, tid);
      reply.send(result);
    }
  );

  fastify.post(
    "/stockBySKU",
    {
      schema: {
        description: "post some data",
        tags: ["berapak"],
        summary: "location based stock",
        body: {
          type: "object",
          properties: {
            skus: {
              type: "array",
              items: {
                type: "string",
                description: "SKU code",
              },
            },
            showroom: {
              type: "string",
              description: "Showroom ID",
            },
          },
        },
      },
    },
    async (req, reply) => {
      const { headers, body } = req;
      const tid = uuidv4();
```

```
      const skus = req.body.skus;
      const showroomId = req.body.showroom || '';

      logger.info({headers, body}, "post-stockBySku:
input", tid);

      reply.headers(corsHeaders);

      if(skus.length <= 0) {
        const result = { status: false, message:
"Invalid Input" };
        logger.error(result, "post-stockBySku: output",
tid);
        return reply.send(result);
      }

      const result = await
GetStockUseCase.getStockRelatedDetailsBySkus(skus,
showroomId, tid);
      reply.send(result);
    }
  );

  fastify.get(
    "/backOrders/:sku",
    {
      schema: {
        description: "get backorder and in transit
quantity info from a sku",
        tags: ["berapak", "back order", "in transit"],
        summary: "back ordere and in transit quantity",
        params: {
          type: "object",
          properties: {
            sku: { type: "string" },
          },
        },
      },
    },
    async (req, reply) => {
      const tid = uuidv4();
```

```
      const sku = req.params.sku;

      logger.info(req.params, "backOrders-get-input",
tid);

      reply.headers(corsHeaders);

      if(!sku) {
        const result = { status: false, message: "sku
empty" };
        logger.error(result, "backOrders-get-output",
tid);
        reply.send(result);
      }

      const result = await
GetInTransitUseCase.getStockBySKU(sku);
      logger.info(result, "backOrders-get-output",
tid);
      reply.send(result);
    }
  );
};
```

Kode Sumber 13 Kode Sumber Interface Stock Data Berapak

```
apiVersion: extensions/v1beta1
metadata:
 name: prod-whatsapp
 namespace: service-prod
 annotations:
   kompose.cmd: kompose convert
 creationTimestamp: null
 labels:
   io.kompose.service: prod-whatsapp
spec:
 template:
   metadata:
     creationTimestamp: null
     labels:
       io.kompose.service: prod-whatsapp
   spec:
     containers:
       - image: registry-intl.ap-southeast-
5.aliyuncs.com/fabelio-services/fabelio-
whatsapp:development-latest
         name: fabelio-whatsapp
         imagePullPolicy: Always
         resources: {}
         envFrom:
           - configMapRef:
               name: fabelio-whatsapp-config
         env:
               - name: DD_AGENT_HOST
                 valueFrom:
                   fieldRef:
                     apiVersion: v1
         command: ["/bin/sh", "-c"]
         args:
           [
             "/usr/local/bin/dumb-init -- node
./dist/src/interfaces/service/EventProcessor.js",
           ]
     imagePullSecrets:
       - name: regsecret
     restartPolicy: Always
status: {}
```

Kode Sumber 14 Kode Sumber CICD Kubernetes Whatsapp

```
import tracer from 'dd-trace';
import logger from './logger';
import config from '../../config';

export default (serviceName) => {
 if (config.get('NODE_ENV') === 'production' ||
config.get('NODE_ENV') === 'staging') {
   tracer.init({
     env: config.get('NODE_ENV'),
     hostname: config.get('DD_AGENT_HOST'),
     port: '8126',
     tags: {
       app: 'Whatsapp',
       service: serviceName,
     },
     analytics: true,
     debug: true,
     logger: {
       debug: (message) => logger.info(message),
       error: (err) => logger.error(err),
     },
   });
 }
};
```

Kode Sumber 15 Kode Sumber Datadog Tracer Whatsapp

```
      httpGet:
        path: /health
        port:3000

      initialDelaySeconds: 20
      timeoutSeconds: 2
      successThreshold: 1
      failureThreshold: 2
      periodSeconds: 15
    readinessProbe:
      httpGet:
        path: /health
        port:3000

      initialDelaySeconds: 5
      timeoutSeconds: 2
      successThreshold: 1
      failureThreshold: 2
      periodSeconds: 5
```

Kode Sumber 16 Kode Sumber CICD PMS

```
const tracer = require('../../helpers/tracer');

tracer.default('API');

const config = require('../../../config');
const fastify = require("fastify")({ logger:
config.NODE_ENV !== 'test' });

const ProductDataApp = require("./ProductData");
const swagger = require('./swagger');
const connect = require('../../infra/mongo').default;

const { PORT } = config.PMS.REST;
const isShutDown = false;

fastify.register(swagger);
fastify.register(ProductDataApp);

fastify.get('/', function(request, reply) {
 reply.send({ service: 'PMS' })
});

const start = async () => {
 console.log('starting');
 try {
   await fastify.listen(PORT, "0.0.0.0");
 } catch (err) {
   fastify.log.error(err);
   process.exit(1);
 }
};

fastify.get('/health', async (request, reply) => {
 if (isShutDown) {
   fastify.log.info('shut down checker');
   return reply.code(503).send('shutting down');
 }

 reply.send('healthy');
});
```

```
connect();
start();

process
  .on('unhandledRejection', (reason, p) => {
    fastify.log.error(reason, 'unhandler rejection at
promise');
  })
  .on('uncaughtException', err => {
    fastify.log.error(err, 'uncaught exception');
  })
  .on('SIGTERM', () => {
    fastify.log.info('SIGTERM sent');
    isShutDown = true;
  });

module.exports = fastify;
```

Kode Sumber 17 Kode Sumber Server PMS

```
<!DOCTYPE html>
<html>
 <head>
   <title>Whatsapp API</title>
   <!-- needed for adaptive design -->
   <meta charset="utf-8"/>
   <meta name="viewport" content="width=device-width,
initial-scale=1">
   <link
href="https://fonts.googleapis.com/css?family=Montserr
at:300,400,700|Roboto:300,400,700" rel="stylesheet">

   <!--
   ReDoc doesn't change outer page styles
   -->
   <style>
     body {
       margin: 0;
       padding: 0;
     }
   </style>
 </head>
 <body>
   <redoc spec-url='openapi.json'></redoc>
   <script
src="https://cdn.jsdelivr.net/npm/redoc@next/bundles/r
edoc.standalone.js"> </script>
 </body>
</html>
```

Kode Sumber 18 Kode Sumber Interface API Documentation Whatsapp

```
import tracer from '../../helpers/tracer';
import 'core-js/stable';
import 'regenerator-runtime/runtime';

tracer('API');

import Fastify from 'fastify';
import Webhook from './webhook';

import path from 'path';
import serveStatic from 'serve-static';
const fastify = Fastify({ logger: true });

fastify.register(Webhook);

fastify.get('/', (request, reply) => {
 reply.send({ service: 'smooch' });
});

fastify.use('/docs', serveStatic(path.join(__dirname,
'../../../docs')));

const start = async () => {
 try {
   await fastify.listen(3000, '0.0.0.0');
   fastify.log.info(`server listening on
${fastify.server.address().port}`);
 } catch (err) {
   fastify.log.error(err);
   process.exit(1);
 }
};

module.exports = fastify;

if (require.main === module) {
 start();
}
```

Kode Sumber 19 Kode Sumber Server API Documentation Whatsapp

BIODATA PENULIS

Nama	: Yovi Agustian
Tempat, Tanggal Lahir	: Bandar Lampung, 15 Agustus 1999
Jenis Kelamin	: Laki-laki
Agama	: Katholik
Status	: Belum Menikah
Alamat	: Jalan Urip Sumoharjo, Sungai 3, Gunung Sulah
Telepon	: 0895377266596
Email	: yoviagustian94@gmail.com

PENDIDIKAN FORMAL

2017 – sekarang	: Mahasiswa S1 Teknik Informatika ITS
2014 – 2017	: SMA Xaverius Bandar Lampung
2011 – 2014	: SMP Xaverius 2 Bandar Lampung
2005 – 2011	: SD Kartika II-5 Bandar Lampung

KEMAMPUAN

- *Web Programming* (Javascript, PHP, HTML)
- *Programming* (C, C++, Python)
- Sistem Operasi (Windows, Ubuntu)
- Bahasa (Indonesia, Inggris)

AKADEMIS

Kuliah	: Departemen Teknik Informatika – Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember Surabaya
Angkatan	: 2017
Semester	: 8 (delapan)
IPK	: 3.62 (Semester 7)